Clustering

Remo Suppi Boldrito

P07/M2103/02290



Índex

Introducció 5

1. Introducción al HPC 7

1.1. Beowulf 8

1.1.1. ¿Cómo configurar los nodos? 9

1.1.2. Beneficios de cómputo distribuido 10

1.2. ¿Cómo hay que programar para aprovechar la concurrencia? 12

1.2.1. PVM, *parallel virtual machine* 13

1.2.2. MPI, *message passing interface* 17

2. OpenMosix 22

3. *Metacomputers, grid computing* 25

3.1. Diferentes arquitecturas de cómputo 25

3.2. Globus 27

3.3. Software, instalación y administración de Globus 29

Actividades 31

Bibliografía 32

GNU Free Documentation License 42

Introducció

Se denomina *cluster* (agrupación) de ordenadores a un grupo de ordenadores que trabajan con un fin común. Estos ordenadores agrupan hardware, redes de comunicación y software para trabajar conjuntamente como si fueran un único sistema. Existen muchas razones atrayentes para realizar estas agrupaciones, pero la principal es poder efectuar el procesamiento de la información de forma más eficiente y rápida como si fuera un único sistema. Generalmente, un *cluster* trabaja sobre una red de área local (LAN) y permite una comunicación eficiente, si bien las máquinas se encuentran dentro de un espacio físico próximo. Una concepción mayor del concepto es la llamada *grid*, donde el objetivo es el mismo, pero implica agrupaciones de ordenadores unidos por redes de área extensa (WAN). Algunos autores consideran el *grid* como un *cluster* de *clusters* en un sentido 'global'. Si bien cada vez más la tecnología y los costes permiten estas aproximaciones, los esfuerzos y la complejidad de utilización de decenas o centenares (en algunos casos, miles) es muy grande. Sin embargo, las ventajas en tiempo de cómputo hacen que, aun así, este tipo de soluciones para el cómputo de altas prestaciones (HPC, *high performance computing*) sean consideradas muy atractivas y en constante evolución. En esta unidad se mostrarán algunas de las aproximaciones más difundidas y utilizadas. [Rad, Dieb, Prob, Prod, Proe, Gloa]

Nota

Un *cluster* es un conjunto de ordenadores en una LAN que trabajan con un objetivo común.

Los *grids* son agrupaciones de ordenadores unidos por redes de área extensa (WAN).

1. Introducción al HPC

Los avances en la tecnología han significado procesadores rápidos, de bajo coste y redes altamente eficientes, lo cual ha favorecido un cambio de la relación precio/prestaciones en favor de la utilización de sistemas de procesadores interconectados en lugar de un único procesador de alta velocidad. Este tipo de arquitectura se puede clasificar en dos configuraciones básicas:

- *Tighly coupled systems:* son sistemas donde la memoria es compartida por todos los procesadores (*shared memory systems*) y la memoria de todos ellos se 've' (por el programador) como una única memoria.

- *Loosely couple systems:* no comparten memoria (cada procesador posee la suya) y se comunican por mensajes pasados a través de una red (*message passing systems*).

En el primer caso son conocidos como sistemas paralelos de cómputo (*parallel processing system*) y en el segundo como sistemas distribuidos de cómputo (*distributed computing systems*). En este último caso podemos decir que un sistema distribuido es una colección de procesadores interconectados por una red donde cada uno tiene sus propios recursos (memoria y periféricos) y se comunican intercambiando mensajes por la red.

La historia de los sistemas informáticos es muy reciente (se puede decir que comienza en la década de los sesenta). En un principio eran sistemas grandes, pesados, caros, de pocos usuarios expertos, no accesibles, lentos. En la década de los setenta, la evolución permitió mejoras sustanciales llevadas a cabo por tareas interactivas (*interactive jobs*), tiempo compartido (*time sharing*), terminales y con una considerable reducción del tamaño. La década de los ochenta se caracteriza por un aumento notable de las prestaciones (hasta hoy en día) y una reducción del tamaño en los llamados *microcomputers*. Su evolución ha sido a través de las estaciones de trabajo (*workstations*) y los avances en redes (LAN de 10 Mbits/s y WAN de 56 Kbytes/s en 1973 a LAN de 1Gbit/s y WAN con ATM, *asynchronous transfer mode* de 1.2 Gbits/s en la actualidad), que es un factor fundamental en las aplicaciones multimedia actuales y de un futuro próximo. Los sistemas distribuidos, por su parte, comenzaron su historia en la década de los setenta (sistemas de 4 u 8 ordenadores) y su salto a la popularidad lo hicieron en la década de los noventa.

Si bien su administración/instalación/mantenimiento es compleja porque continúan creciendo, las razones básicas de su popularidad son el incremento de prestaciones que presentan en aplicaciones intrínsecamente distribuidas (por su naturaleza), la información compartida por un conjunto de usuarios,

compartir recursos, la alta tolerancia a los fallos y la posibilidad de expansión incremental (capacidad de agregar más nodos para aumentar las prestaciones de modo incremental).

En los próximos apartados veremos algunos de los sistemas más comunes de procesamiento paralelo/distribuido, así como los modelos de programación utilizados para generar código capaz de utilizar estas prestaciones.

1.1. Beowulf

Beowulf [Rad, Beo] es una arquitectura multiordenador que puede ser utilizada para aplicaciones paralelas/distribuidas (APD). El sistema consiste básicamente en un servidor y uno o más clientes conectados (generalmente) a través de Ethernet y sin la utilización de ningún hardware específico. Para explotar esta capacidad de cómputo, es necesario que los programadores tengan un modelo de programación distribuido que, si bien a través de UNIX es posible (socket, rpc), puede significar un esfuerzo considerable, ya que son modelos de programación a nivel de *systems calls* y lenguaje C, por ejemplo; pero este modo de trabajo puede ser considerado de bajo nivel.

La capa de software aportada por sistemas tales como *parallel virtual machine* (PVM) y *message passing interface* (MPI) facilita notablemente la abstracción del sistema y permite programar APD de modo sencillo y simple. La forma básica de trabajo es maestro-trabajadores (*master-workers*), en que existe un servidor que distribuye la tarea que realizarán los trabajadores. En grandes sistemas (por ejemplo, de 1.024 nodos) existe más de un maestro y nodos dedicados a tareas especiales como, por ejemplo, entrada/salida o monitorización.

Nota

Varias opciones:
- Beowulf
- OpenMosix
- Grid (Globus)

Una de las principales diferencias entre beowulf y un *cluster of workstations* (COW) es que Beowulf se 've' como una única máquina donde los nodos se acceden remotamente, ya que no disponen de terminal (ni de teclado), mientras que un COW es una agrupación de ordenadores que pueden ser utilizados tanto por los usuarios de la COW, como por otros usuarios en forma interactiva a través de su pantalla y teclado. Hay que considerar que Beowulf no es un software que transforma el código del usuario en distribuido ni afecta al *kernel* del sistema operativo (como por ejemplo Mosix). Simplemente, es una forma de agrupación (*cluster*) de máquinas que ejecutan GNU/Linux y actúan como un superordenador. Obviamente, existe gran cantidad de herramientas que permiten obtener una configuración más fácil, bibliotecas o modificaciones al *kernel* para obtener mejores prestaciones, pero es posible construir un *cluster* Beowulf a partir de un GNU/Linux estándar y de software convencional. La construcción de un *cluster* beowulf de dos nodos, por ejemplo, se puede llevar a cabo simplemente con las dos máquinas conectadas por Ethernet mediante un *hub*, una distribución de GNU/ Linux estándar (Debian), el sistema de archivos compartido (NFS) y tener habilitados los servicios de red como rsh

o ssh. En estas condiciones, se puede argumentar que se dispone de un *cluster* simple de dos nodos.

1.1.1. ¿Cómo configurar los nodos?

Primero se debe modificar de (cada nodo) el /etc/hosts para que la línea de localhost sólo tenga el 127.0.0.1 y no incluya ningún nombre de la máquina, por ejemplo:

127.0.0.1 localhost

Y añadir las IP de los nodos (y para todos los nodos), por ejemplo:

192.168.0.1 pirulo1
192.168.0.2 pirulo2
...

Se debe crear un usuario (nteum) en todos los nodos, crear un grupo y añadir este usuario al grupo:

```
groupadd beowulf
adduser nteum beowulf
echo umask 007 >> /home/nteum/.bash_profile
```

Así, cualquier archivo creado por el usuario nteum o cualquiera dentro del grupo será modificable por el grupo beowulf.

Se debe crear un servidor de NFS (y los demás nodos serán clientes de este NFS). Sobre el servidor hacemos:

mkdir /mnt/nteum
chmod 770 /mnt/nteum
chown -R nteum:beowulf /mnt/nteum

Ahora exportamos este directorio desde el servidor.

```
cd /etc
cat >> exports
```

/mnt/wolf 192.168.0.100/192.168.0.255 (rw)
<control d>

Debemos tener en cuenta que nuestra red será 192.168.0.xxx y es una red privada, es decir, el *cluster* no se verá desde Internet y deberemos ajustar las configuraciones para que todos los nodos se vean entre todos (desde los *firewalls*).

Verificamos que los servicios están funcionando:

```
chkconfig -add sshd
chkconfig -add nfs
```

```
chkconfig -add rexec
chkconfig -add rlogin
chkconfig -level 3 rsh on
chkconfig -level 3 nfs on
chkconfig -level 3 rexec on
chkconfig -level 3 rlogin on
```

Para trabajar de manera segura es importante trabajar con ssh en lugar de rsh, por lo cual deberíamos generar las llaves para interconectar de modo seguro las máquinas-usuario nteum sin passwd. Para eso modificamos (quitamos el comentario #), de /etc/ssh/sshd_config en las siguientes líneas:

RSAAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys

Reiniciamos la máquina y nos conectamos como usuario nteum, ya que el *cluster* lo operará este usuario. Para generar las llaves:

```
ssh-keygen -b 1024 -f ~/.ssh/id_rsa -t rsa -N ""
```

En el directorio /home/nteum/.ssh se habrán creado los archivos id_rsa and id_rsa.pub y se debe copiar id_rsa.pub en un archivo llamado authorized_keys en el mismo directorio. Y modificamos los permisos con `chmod 644 ~/.ssh/aut* y chmod 755 ~/.ssh`.

Ya que sólo el nodo principal se conectará a todos los restantes (y no a la inversa) necesitamos copiar únicamente la llave pública (d_rsa.pub) a cada nodo en el directorio/archivo /home/nteum/.ssh/authorized_keys de cada nodo. Sobre cada nodo además se deberá montar el NFS agregando /etc/fstab la línea pirulo1:/mnt/nteum /mnt/nteum nfs rw,hard,intr 0 0.

A partir de aquí ya tenemos un *cluster* beowulf para ejecutar aplicaciones que podrán ser PVM o MPI (ya se verá en los siguientes apartados). Sobre FC existe una aplicación (system-config-cluster) que permite configurar un *cluster* en base a una herramienta gráfica. Para más información, consultar: http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.html.

1.1.2. Beneficios de cómputo distribuido

¿Dónde están los beneficios del cómputo paralelo? Lo veremos con un ejemplo [Rad]. Sea un programa para sumar números (por ejemplo, 4 + 5 + 6...) llamado sumdis.c y escrito en C:

```
#include <stdio.h>

   int main (int argc, char** argv){

float inicial, final, resultado, tmp;

if (argc < 2) {
   printf (“Uso: %s N.° inicial N.° final\n”,argv[0]);
   exit(1);
   }
else {
   inicial = atol (argv[1]);
   final = atol (argv[2]);
   resultado = 0.0;
   }
   for (tmp = inicial; tmp <= final; tmp++){
   resultado + = tmp; }
printf(“%f\n”, resultado)
return 0;
}
```

Lo compilamos con `gcc -o sumdis sumdis.c` y si miramos la ejecución de este programa con, por ejemplo:

`time ./sumdis 1 1000000` (desde 1 a 106)

se podrá observar que el tiempo en una máquina Debian 2.4.18 con AMD athjon 1.400 MHz 256 Mb RAM es (aproximadamente) real = 0,013 y user = 0,010 es decir, 13 ms en total y 10 ms en zona de usuario. Si, en cambio, hacemos:

`time ./suma 1 16000000` (desde 1 a 16 * 106)

el tiempo será real = 182, es decir, 14 veces más, lo cual, si se considera 160.000.000 ($160*10^6$), el tiempo será del orden de decenas de minutos.

La idea del cómputo distribuido es: si disponemos de un *cluster* de 4 máquinas (nodo1-nodo4) con un servidor, donde el fichero se comparte por NFS, sería interesante dividir la ejecución a través de rsh (no recomendable, pero como ejemplo es aceptable), de modo que el primero sume de 1 a 40.000.000, el segundo de 40.000.001 a 80.000.000, el tercero de 80.000.001 a 120.000.000 y el cuarto de 120.000.001 a 160.000.000. Los siguientes comandos muestran una posibilidad. Consideramos que el sistema tiene el directorio /home compartido por NFS y el usuario (nteum) que ejecutará el *script* tiene adecuadamente configurado el .rhosts para acceder sin contraseña a su cuenta. Además, si se ha activado el tcpd en /etc/inetd.conf en la línea de *rsh*, debe existir la correspondiente en el /etc/hosts.allow, que permita acceder a las cuatro máquinas del *cluster*:

`mkfifo salida` Crea una cola fifo en /home/nteum

```
./distr.sh & time cat salida | awk '{total + = $1 } \
END printf “%lf”, total}'
```

Ejecuta el comando distr.sh; se recolectan los resultados y se suman mientras se mide el tiempo de ejecución

El *shell script* distr.sh puede ser algo como:

```
rsh nodo1 /home/nteum/sumdis 1 40000000 > /home/nteum/salida < /dev/null &
rsh nodo2 /home/nteum/sumdis 40000001 80000000 > /home/nteum/salida < /dev/null &
rsh nodo3 /home/nteum/sumdis 80000001 120000000 > /home/nteum/salida < /dev/null &
rsh nodo4 /home/nteum/sumdis 120000001 160000000 > /home/nteum/salida < /dev/null &
```

Podremos observar que el tiempo se reduce notablemente (aproximadamente en un valor cercano a 4) y no exactamente en forma lineal, pero muy próxima. Obviamente, este ejemplo es muy simple y sólo con fines demostrativos. Los programadores utilizan bibliotecas que les permiten realizar el tiempo de ejecución, la creación y comunicación de procesos en un sistema distribuido (por ejemplo, PVM y MPI).

1.2. ¿Cómo hay que programar para aprovechar la concurrencia?

Hay varias maneras de expresar la concurrencia en un programa. Las dos más comunes son:

1) Utilizando *threads* (o procesos).

2) Utilizando procesos en diferentes procesadores que se comunican por mensajes (MPS, *message passing system*).

Ambos métodos pueden ser implementados sobre diferentes configuraciones hardware (memoria compartida o mensajes), pero los sistemas MPS presentan el problema de latencia y velocidad de los mensajes en la red, lo que puede resultar un factor negativo. Sin embargo, con el avance de las tecnologías de red estos sistemas han crecido en popularidad (y en cantidad). Un mensaje es sumamente simple:

send(*destino*,*msg*)
recv(*origen*,*msg*)

Las API más comunes hoy en día son PVM y MPI y además no limitan la posibilidad de utilizar *threads* (aunque a nivel local) y tener concurrencia entre procesamiento y entrada/salida. En cambio, en una máquina de memoria compartida (SHM, *shared memory*) sólo es posible utilizar *threads* y tiene el problema grave de la escalabilidad, ya que todos los procesadores utilizan la misma memoria y el número de procesadores en el sistema está limitado por el ancho de banda de la memoria.

En resumen, podemos concluir:

1) Proliferación de máquinas multitareas (multiusuario) conectadas por red con servicios distribuidos (NFS y NIS YP).

2) Son sistemas heterogéneos con sistemas operativos de tipo NOS (*networked operating system*) que ofrecen una serie de servicios distribuidos y remotos.

3) La programación de aplicaciones distribuidas se puede efectuar a diferentes niveles:

a) Utilizando un modelo cliente-servidor y programando a bajo nivel (*sockets*).

b) El mismo modelo, pero con API de "alto" nivel (PVM, MPI).

c) Utilizando otros modelos de programación como, por ejemplo, programación orientada a objetos distribuidos (RMI, CORBA, Agents...).

1.2.1. PVM, *parallel virtual machine*

PVM [Proe] es una API que permite generar, desde el punto de vista de la aplicación, una colección dinámica de ordenadores, que constituyen una máquina virtual (VM). Las tareas pueden ser creadas dinámicamente (*spawned*) y/o eliminadas (*killed*) y cualquier tarea PVM puede enviar un mensaje a otra. No existe un límite en el tamaño o número de mensajes (según las especificaciones, aunque pueden existir combinaciones hardware/sistema operativo que den como resultado limitaciones en el tamaño del mensaje) y el modelo soporta: tolerancia a fallos, control de recursos, control de procesos, heterogeneidad en las redes y en los *hosts*.

El sistema (VM) dispone de herramientas para el control de recursos (agregar o quitar *hosts* de la máquina virtual), control de procesos (creación/eliminación dinámica de procesos), diferentes modelos de comunicación (*blocking send*, *blocking*/*nonblocking receive*, *multicast*), grupos de tareas dinámicos (una tarea puede anexarse a un grupo o no dinámicamente) y tolerancia a fallos (la VM detecta el fallo y se puede reconfigurar).

La estructura de PVM se basa por un lado en el *daemon* (pvm3d) que reside en cada máquina y se interconectan utilizando UDP, y por el otro, la biblioteca de PVM (libpvm3.a), que contiene todas las rutinas para enviar/recibir mensajes, crear/eliminar procesos, grupos, sincronización, etc. y que utilizará la aplicación distribuida.

PVM dispone de una consola (pvm) que permite poner en marcha el *daemon*, crear la VM, ejecutar aplicaciones, etc. Es recomendable instalar el software desde la distribución, ya que su compilación requiere cierta 'dedicación'. Para instalar PVM sobre Debian, por ejemplo, se deben incluir dos paquetes (mínimo): pvm y pvm-dev (en el primero está la consola pvm y utilidades y en el segundo librerías, *header* y el resto de las herramientas para compilar). Si sólo

se necesita la librería porque ya se tiene la aplicación se puede instalar únicamente el paquete libpvm3).

Para realizar una aplicación paralela/distribuida en PVM, se puede partir de la versión serie o mirando la estructura física del problema y determinar qué partes pueden ser concurrentes (independientes). Las partes concurrentes serán candidatas a reescribirse como código paralelo. Además, se debe considerar si es posible reemplazar las funciones algebraicas por sus versiones paralelizadas (por ejemplo, ScaLapack, Scalable Linear Algebra Package, disponibles en Debian como scalapack-pvm | mpich-test | dev, scalapack1-pvm | mpich según sean para PVM o MPI). También es conveniente averiguar si hay alguna aplicación similar paralela (http://www.epm.ornl.gov/pvm) que pueda orientarnos sobre el modo de construcción de la aplicación paralela.

Paralelizar un programa no es una tarea fácil, ya que se debe tener en cuenta la ley de Amdahl.

> La ley de Amdahl afirma que el incremento de velocidad (speedup) está limitado por la fracción de código (f) que puede ser paralelizado: speedup = 1/(1-f).

Esta ley implica que una aplicación secuencial f = 0 y el speedup = 1, con todo el código paralelo f = 1 y speedup = infinito(!), con valores posibles, 90% del código paralelo significa un speedup = 10 pero con f = 0,99 el speedup = 100. Esta limitación se puede evitar con algoritmos escalables y diferentes modelos de aplicación:

Nota

Ley de Amdahl:
speedup = 1/(1-f)
f es la fracción código paralelo

1) Maestro-trabajador: el maestro inicia a todos los trabajadores y coordina su trabajo y entrada/salida.

2) *Single process multiple data* (SPMD): mismo programa que se ejecuta con diferentes conjuntos de datos.

3) Funcional: varios programas que realizan una función diferente en la aplicación.

Con la consola pvm y a través del comando add se puede configurar la VM añadiendo todos los nodos. En cada uno de ellos debe existir el directorio ~/pvm3/bin/LINUX con los binarios de la aplicación. Deben estar declaradas la variables *PVM_ROOT* = Directorio, donde se encuentra lib/LINUX/libpvm3.a y PVM_ARCH=LINUX, que se pueden poner, por ejemplo, en el archivo /.cshrc. El *shell* por defecto del usuario (generalmente un usuario *NIS* o si no, en cada máquina debe existir el mismo usuario con la misma contraseña) debe ser el csh (si utilizamos el rsh como medio de ejecución remota) y debe estar configurado el archivo /.rhosts para permitir el acceso sin *password* a cada nodo. El paquete

PVM incorpora un rsh-pvm que se puede encontrar en /usr/lib/pvm3/bin como rsh específico para PVM (ver la documentación), ya que existen algunas distribuciones que no lo incluyen por motivos de seguridad. Es recomendable configurar, como antes se ha mostrado, el ssh con las llaves públicas del servidor en .ssh/autorized_keys del directorio de cada usuario.

Como ejemplo de programación PVM, se muestra un programa del tipo servidor-clientes, donde el servidor crea los hijos, les envía datos, éstos circulan los datos una determinada cantidad de veces entre los hijos (el de la izquierda recibe un dato, lo procesa y se lo envía al de la derecha) mientras el padre espera que cada hijo termine.

Ejemplo en PVM: master.*c*

Para **compilar** en Debian:
gcc -O -I/usr/share/pvm3/include/ -L/usr/share/pvm3/lib/LINUX -o master master.c -lpvm3
Los directorios en -I y en -L deben ser donde se encuentran los includes pvm3.h y libpvm* respectivamente.
Ejecución:
1) ejecutar el daemon pvmd con `pvm`
2) ejecutar add para agregar los nodos (este comando se puede saltar si solo se tiene un nodo
3) ejecutar quit (salimos del pvm pero queda ejecutándose)
4) ejecutamos master

Nota

Compilar PVM:
gcc -O -I/usr/include/ -o output output.c -lpvm3

```
#include <stdio.h>
#include "pvm3.h"
#define SLAVENAME "/home/nteum/pvm3/cliente"
main() {
 int mytid, tids[20], n, nproc, numt, i, who, msgtype, loops;
 float data[10]; int n_veces;

 if( pvm_parent() ==PvmNoParent ){
      /*Retorna si es el proceso padre o hijo*/
      loops = 1;
      printf("\n Cuántos hijos (120)? ");
      scanf("%d", &nproc);
      printf("\n Cuántos bucles de comunicación hijo-hijo (1 - 5000)? "); scanf("%d", &loops); }

 /*Redirecciona la entrada/salida de los hijos al padre*/
 pvm_catchout(stdout);

 /*Crea los hijos*/
 numt = pvm_spawn(SLAVENAME, (char**)0, 0, "", nproc, tids);
 /*Inicia un nuevo proceso, 1.° :ejecutable hijo, 2.° : argv, 3.° :opciones,
 4.° :donde, 5.° :N.° copias, 6.° :matriz de id*/
 printf("Resultado del Spawn: %d \n", numt);

 /*Ha podido?*/
 if( numt < nproc ){
      printf("Error creando los hijos. Código de error:\n");
      for( i = numt ; i<nproc ; i++ ) {
      printf("Tid %d %d\n",i,tids[i]); }
      for( i = 0 ; i<numt ; i++ ){
      pvm_kill( tids[i] ); }        /*Mata los procesos con id en tids*/
      pvm_exit();
      exit(); /*Termina*/
 }

 /*Inicio del programa de padre, inicializa los datos*/
 n = 10;
 for( i = 0 ; i<n ; i++ ){
      data[i] = 2.0;}
 /*Broadcast con datos iniciales a los slaves*/
 pvm_initsend(PvmDataDefault);.
```

```
/*Borra el buffer y especifica el message encoding*/
pvm_pkint(&loops, 1, 1);
/*Empaqueta un dato en el buffer, 2.° N.°, 3*:stride*/
pvm_pkint(&nproc, 1, 1);
pvm_pkint(tids, nproc, 1);
pvm_pkint(&n, 1, 1);
pvm_pkfloat(data, n, 1);
pvm_mcast(tids, nproc, 0);
/*Multicast en el buffer a los tids y espera el resultado de los hijos */
msgtype = 5;
for( i = 0 ; i < nproc ; i++ ){
     pvm_recv( -1, msgtype );
     /*Recibe un mensaje, -1 :de cualquiera, 2.ª :tag del msg*/
     pvm_upkint( &who, 1, 1 );
     /*Desempaqueta*/
     printf("Terminó %d\n",who);
     }
pvm_exit();
}
```

Ejemplo de PVM: *cliente.c*

Para **compilar** en Debian:
gcc -O -I/usr/share/pvm3/include/ -L/usr/share/pvm3/lib/LINUX -o cliente cliente.c -lpvm3
Los directorios en -I y en -L deben ser donde se encuentran los includes pvm3.h y libpvm* respectivamente.
Ejecución:
No es necesario ya que los pondrá en marcha el master pero el cliente debe estar en /home/nteum/pvm3

```
#include <stdio.h>
#include "pvm3.h"

main() {
 int mytid;/*Mi task id*/
 int tids[20];/*Task ids*/
 int n, me, i, nproc, master, msgtype, loops; float data[10];
 long result[4]; float work();
 mytid = pvm_mytid(); msgtype = 0;

 pvm_recv( -1, msgtype );
 pvm_upkint(&loops, 1, 1);
 pvm_upkint(&nproc, 1, 1);
 pvm_upkint(tids, nproc, 1);
 pvm_upkint(&n, 1, 1);
 pvm_upkfloat(data, n, 1);
 /*Determina qué hijo es (0 -- nproc-1) */
 for( i = 0; i < nproc ; i++ )
 if( mytid == tids[i] ){ me = i; break; }

 /*Procesa y pasa los datos entre vecinos*/
 work( me, data, tids, nproc, loops );

 /*Envía los datos al máster */
 pvm_initsend( PvmDataDefault );
 pvm_pkint( &me, 1, 1 );
 msgtype = 5;
 master = pvm_parent();      /*Averigua quién lo creó*/
 pvm_send( master, msgtype);
 pvm_exit();
 }

float work(me, data, tids, nproc, loops)
 int me, *tids, nproc; float *data; {
   int i,j, dest; float psum = 0.0, sum = 0.1;
   for (j = 1; j <= loops; j++){
      pvm_initsend( PvmDataDefault );
      pvm_pkfloat( &sum, 1, 1 );
      dest = me + 1;
      if( dest == nproc ) dest = 0;
      pvm_send( tids[dest], 22 );
```

```
        i = me - 1;
        if (me == 0 ) i = nproc-1;
        pvm_recv( tids[i], 22 );
        pvm_upkfloat( &psum, 1, 1 );
        }
    }
```

El programador cuenta con la gran ayuda de una interfaz gráfica (ver figura siguiente) que actúa como consola y monitor de PVM llamada xpvm (en Debian XPVM, instalar paquete xpvm), que permite configurar la VM, ejecutar procesos, visualizar la interacción entre tareas (comunicaciones), estados, información, etc.

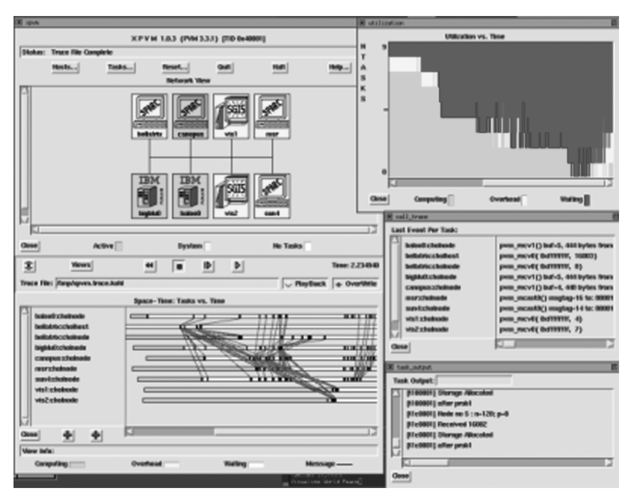


Figura 1

1.2.2. MPI, *message passing interface*

La definición de la API de MPI [Prob, Proc] ha sido el trabajo resultante del MPI Fourum (MPIF), que es un consorcio de más de 40 organizaciones. MPI tiene influencias de diferentes arquitecturas, lenguajes y trabajos en el mundo del paralelismo como son: WRC (Ibm), Intel NX/2, Express, nCUBE, Vertex, p4, Parmac y contribuciones de ZipCode, Chimp, PVM, Chamaleon, PICL. El principal objetivo de MPIF fue diseñar una API, sin relación particular con ningún compilador ni biblioteca, de modo que permitiera la comunicación eficiente (*memory-to-memory copy*), cómputo y comunicación concurrente y descarga de comunicación, siempre y cuando exista un coprocesador de comunicaciones. Además, que soportara el desarrollo en ambientes heterogéneos, con interfaz C y F77 (incluyendo C++, F90), donde la comunicación fuera fiable y los fallos resueltos por el sistema. La API también debía tener interfaz para diferentes entornos (PVM, NX, Express, p4...), disponer una implementación adaptable a diferentes plataformas con cambios insignificantes y que no interfiera con el sistema operativo (*thread-safety*). Esta API fue diseñada especialmente para programadores que utilizaran el *message passing paradigm* (MPP) en C y F77 para aprovechar la característica más relevante: la portabilidad. El MPP se puede ejecutar sobre máquinas multiprocesadores, redes de WS e incluso sobre máquinas de memoria compartida. La versión MPI1 (la versión

más extendida) no soporta creación (*spawn*) dinámica de tareas, pero MPI2 (en creciente evolución) sí que lo hace.

Muchos aspectos han sido diseñados para aprovechar las ventajas del hardware de comunicaciones sobre SPC (*scalable parallel computers*) y el estándar ha sido aceptado mayoritariamente por los fabricantes de hardware paralelo y distribuido (SGI, SUN, Cray, HPConvex, IBM, Parsystec...). Existen versiones *freeware* (por ejemplo, mpich) (que son totalmente compatibles con las implementaciones comerciales realizadas por los fabricantes de hardware) e incluyen comunicaciones punto a punto, operaciones colectivas y grupos de procesos, contexto de comunicaciones y topología, soporte para F77 y C y un entorno de control, administración y *profiling*. Pero existen también algunos puntos no resueltos como son: operaciones SHM, ejecución remota, herramientas de construcción de programas, depuración, control de *threads*, administración de tareas, funciones de entrada/salida concurrentes (la mayor parte de estos problemas de falta de herramientas están resueltos en la versión 2 de la API MPI2). El funcionamiento en MPI1, al no tener creación dinámica de procesos, es muy simple, ya que de tantos procesos como tareas existan, autónomos y ejecutando su propio código estilo MIMD (*multiple instruction multiple data*) y comunicándose vía llamadas MPI. El código puede ser secuencial o *multithread* (concurrentes) y MPI funciona en modo *threadsafe*, es decir, se pueden utilizar llamadas a MPI en *threads* concurrentes, ya que las llamadas son reentrantes.

Para la instalación de MPI se recomienda utilizar la distribución, ya que su compilación es sumamente compleja (por las dependencias que necesita de otros paquetes). Debian incluye la versión Mpich 1.2.7-2 (Etch) en el paquete mpich-bin (el de mpich es obsoleto) y también mpich-mpd-bin (versión de *multipurpose daemon*, que incluye soporte para la gestión y control de procesos en forma escalable). El mpich-bin implementa el estándar MPI 1.2 y algunas partes de MPI 2 (como, por ejemplo, entrada/salida paralela). Además, esta misma distribución incluye otra implementación de MPI llamada LAM (paquetes lam* y documentación en /usr/doc/lam-runtime/release.html). Las dos implementaciones son equivalentes desde el punto de vista de MPI, pero se gestionan de forma diferente. Toda la información sobre Mpich se puede encontrar (después de instalar los paquetes mpich*) en /usr/share/doc/mpi (o en /usr/doc/mpi). Mpich necesita rsh para ejecutarse en otras máquinas, lo cual significa insertar en el directorio del usuario un archivo ~/.rhosts con líneas del siguiente formato: *host username* para permitir a *username* entrar en *host* sin *password* (al igual que PVM). Se debe tener en cuenta que hay que instalar el paquete rshserver sobre todas la máquinas y si se tiene tcpd en /etc/inetd.conf sobre *rsh.d*, se debe habilitar los *hosts* en /etc/hosts.allow. Además, se deberá tener montado el directorio del usuario por NFS en todas las máquinas y en el fichero /etc/mpich/machines.LINUX se deberá poner el nombre (*hostname*) de todas las máquinas que forman el *cluster* (una máqui-

na por línea, por defecto aparece *localhost*). Además, el usuario deberá tener como *shell* por defecto el Csh.

Sobre Debian se puede instalar el paquete *update-cluster* para ayudar en su administración. La instalación sobre Debian de Mpich utiliza ssh en lugar de rsh por motivos de seguridad, si bien existe un enlace de rsh =>ssh por compatibilidad. La única diferencia es que se deberán utilizar los mecanismos de validación de ssh para la conexión sin contraseña a través de los ficheros correspondientes. De lo contrario, por cada proceso que se ejecute se deberá introducir la contraseña antes de la ejecución. Para permitir la conexión entre máquinas sin contraseña con ssh, se deberá hacer como se explicó en el apartado anterior. Para probar, se puede hacer ssh *localhost* y se debe poder entrar sin contraseña. Tener en cuenta que si se instala Mpich y LAM-MPI, el mpirun de Mpich se llamará mpirun.mpich y el mpirun será el de LAM-MPI. Es importante recordar que mpirun de LAM utilizará el *daemon* lamboot para formar la topología distribuida de la VM.

El *daemon* lamboot ha sido diseñado para que los usuarios puedan ejecutar programas distribuidos sin tener permisos de *root* (también permite ejecutar programas en una VM sin llamadas a MPI). Por ello, para ejecutar el mpirun se deberá hacer como un usuario diferente de *root* y ejecutar antes lamboot. El lamboot utiliza un archivo de configuración en /etc/lam para la definición por defecto de los nodos (ver bhost*) y consultar la documentación para mayor información (http://www.lam-mpi.org/). [Lam]

Para compilar programas MMPI, se puede utilizar el comando mpicc (por ejemplo, `mpicc -o test test.c`), que acepta todas las opciones de gcc aunque es recomendable utilizar (con modificaciones) algunos de los *makefiles* que se hallan en los ejemplos /usr/doc/mpich/examples. También se puede utilizar mpireconfig Makefile, que utiliza como entrada el archivo Makefile.in para generar el *makefile* y es mucho más fácil de modificar. Después se podrá hacer:

```
mpirun -np 8 programa
```

o bien:

```
mpirun.mpich -np 8 programa
```

donde np es el número de procesos o procesadores en que se ejecutará el programa (8, en este caso). Se puede poner el número que se desee, ya que Mpich intentará distribuir los procesos en forma equilibrada entre todas las máquinas de /etc/mpich/machines.LINUX. Si hay más procesos que procesadores, Mpich utilizará las características de intercambio de tareas de GNU/Linux para simular la ejecución paralela. En Debian y en el directorio /usr/doc/mpich-doc (un

enlace a /usr/share/doc/mpich-doc) se encuentra toda la documentación en diferentes formatos (comandos, API de MPI, etc.).

Para compilar MPI: `mpicc -O -o output output.c`

Ejecutar Mpich: `mpirun.mpich -np N°_procesos output`

A continuación veremos dos ejemplos (que se incluyen con la distribución Mpich 1.2.x en el directorio /usr/doc/mpich/examples). Srtest es un programa simple para establecer comunicaciones entre procesos punto a punto, y cpi calcula el valor de Pi forma distribuida (por integración).

Comunicaciones punto a punto: `srtest.c`

Para compilar: mpicc -O -o srtest srtest.c
Ejecución Mpich: mpirun.mpich -np *N.º_procesos* srtest (solicitará la contraseña [*N.º procesos* - 1] veces si no se tiene el acceso directo por *ssh*).
Ejecución LAM: mpirun -np N.º_procesos srtest (debe ser un usuario diferente de *root*)

```
#include "mpi.h"
#include <stdio.h>
#define BUFLEN 512
int main(int argc, char *argv[]) {
   int myid, numprocs, next, namelen;
   char buffer[BUFLEN], processor_name[MPI_MAX_PROCESSOR_NAME]; MPI_Status status;
   MPI_Init(&argc,&argv);
      /* Debe ponerse antes de otras llamadas MPI, siempre */
      MPI_Comm_size(MPI_COMM_WORLD,&numprocs); MPI_Comm_rank(MPI_COMM_WORLD,&myid);
      /*Integra el proceso en un grupo de comunicaciones*/
      MPI_Get_processor_name(processor_name,&namelen);
      /*Obtiene el nombre del procesador*/
   fprintf(stderr,"Proceso %d sobre %s\n", myid, processor_name); strcpy(buffer,"Hola Pueblo");
   if (myid ==numprocs1) next = 0;
   else next = myid+1;
   if (myid ==0) {/*Si es el inicial, envía string de buffer*/.
      printf("%d Envío '%s' \n",myid,buffer);
      MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR, next, 99,
      MPI_COMM_WORLD);
      /*Blocking Send, 1 o :buffer, 2 o :size, 3 o :tipo, 4 o :destino, 5
      o :tag, 6 o :contexto*/
      /*MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR,
      MPI_PROC_NULL, 299,MPI_COMM_WORLD);*/
      printf("%d recibiendo \n",myid);
      /* Blocking Recv, 1 o :buffer, 2 o :size, 3 o :tipo, 4 o :fuente, 5
      o :tag, 6 o :contexto, 7 o :status*/
      MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99,
      MPI_COMM_WORLD,&status); printf("%d recibió '%s' \n",myid,buffer) }
   else {
      printf("%d recibiendo \n",myid);
      MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99,
      MPI_COMM_WORLD,status);
      /*MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_PROC_NULL, 299,MPI_COMM_WORLD,&status);*/
      printf("%d recibió '%s' \n",myid,buffer);
      MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR, next, 99,
      MPI_COMM_WORLD);
      printf("%d envió '%s' \n",myid,buffer);}
   MPI_Barrier(MPI_COMM_WORLD); /*Sincroniza todos los procesos*/
   MPI_Finalize(); /*Libera los recursos y termina*/ return (0);
   }
```

Cálculo de PI distribuido: `cpi.c`

Para compilar: *mpicc O* o *cpi cpi.c*.
Ejecución Mpich: mpirun.mpich -np N.º procesos cpi (solicitará la contraseña (N.º procesos - 1) veces si no se tiene el acceso directo por *ssh*).
Ejecución LAM: mpirun -np N.º procesos cpi (debe ser un usuario diferente de *root*).

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
double f( double );
double f( double a) { return (4.0 / (1.0 + a*a)); }

int main( int argc, char *argv[] ) {
   int done = 0, n, myid, numprocs, i;
   double PI25DT = 3.141592653589793238462643;
   double mypi, pi, h, sum, x;
   double startwtime = 0.0, endwtime;
   int namelen;
   char processor_name[MPI_MAX_PROCESSOR_NAME];

   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
         /*Indica el número de procesos en el grupo*/
         MPI_Comm_rank(MPI_COMM_WORLD,&myid); /*Id del proceso*/
         MPI_Get_processor_name(processor_name,&namelen);
         /*Nombre del proceso*/
   fprintf(stderr,"Proceso %d sobre %s\n", myid, processor_name);
   n = 0;
   while (!done) {
      if (myid ==0) { /*Si es el primero...*/
         if (n ==0) n = 100; else n = 0;
         startwtime = MPI_Wtime();} /* Time Clock */
      MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD); /*Broadcast al resto*/ /*Envía desde el 4.°
      arg. a todos
      los procesos del grupo Los restantes que no son 0
      copiarán el buffer desde 4 o arg -proceso 0-*/ /*1.°:buffer,
      2.° :size, 3.° :tipo, 5.° :grupo */
      if (n == 0) done = 1; else {
         h = 1.0 / (double) n;
         sum = 0.0;
         for (i = myid + 1; i <= n; i + = numprocs) {
            x = h * ((double)i - 0.5); sum + = f(x); }
         mypi = h * sum;
      MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
      MPI_COMM_WORLD);
      /* Combina los elementos del Send Buffer de cada proceso del
      grupo usando la operación MPI_SUM y retorna el resultado en
      el Recv Buffer. Debe ser llamada por todos los procesos del
      grupo usando los mismos argumentos*/ /*1.° :sendbuffer, 2.°
      :recvbuffer, 3.° :size, 4.° :tipo, 5.° :oper, 6.° :root, 7.°
      :contexto*/
      if (myid == 0){ /*Sólo el P0 imprime el resultado*/
      printf("Pi es aproximadamente %.16f, el error es %.16f\n", pi,   fabs(pi - PI25DT));
      endwtime = MPI_Wtime();
      printf("Tiempo de ejecución = %f\n", endwtime-startwtime); }
      }
   }
   MPI_Finalize(); /*Libera recursos y termina*/
   return 0;
}
```

Al igual que en PVM existe XPVM, en MPI existe una aplicación análoga (más sofisticada) llamada XMPI (en Debian xmpi). También es posible instalar una biblioteca, libxmpi3, que implementa el protocolo XMPI para analizar gráficamente programas MPI con más detalles que los ofrecidos por xmpi. La figura siguiente muestra unas de las posibles gráficas de xmpi.

2. OpenMosix

OpenMosix [Prod] es un paquete software que transforma un conjunto de máquinas conectadas por red bajo GNU/Linux en un *cluster*. Éste equilibra la carga automáticamente entre los diferentes nodos del *cluster* y los nodos pueden unirse o dejar el *cluster* sin interrumpir el servicio. La carga se distribuye entre los nodos teniendo en cuenta la velocidad de la conexión y la CPU. OpenMosix forma parte del *kernel* (a través de un Linux Kernel Patch) y mantiene total compatibilidad con GNU/Linux, los programas de usuario, archivos y recursos. Otra característica de OpenMosix es que incorpora un potente y optimizado sistema de archivos (oMFS) para aplicaciones de HPC (high performance computing). En Debian Woody se puede instalar OpenMosix desde openmosix-dev (bibliotecas y headers), kernel-pacth- openmosix (OpenMosix patch), openmosix (herramientas de administración). También de manera análoga se puede instalar mosix (ver la documentación para las diferencias, sobre todo de licencias, entre Mosix y OpenMosix). En Debian posteriores a Woody no se incluye como paquete (stable) y se debería recurrir a http://openmosix.sourceforge.net/ para obtener los paquetes (o fuentes) y las guías de instalación (http://howto.x-tend.be/openMosix-HOWTO/).

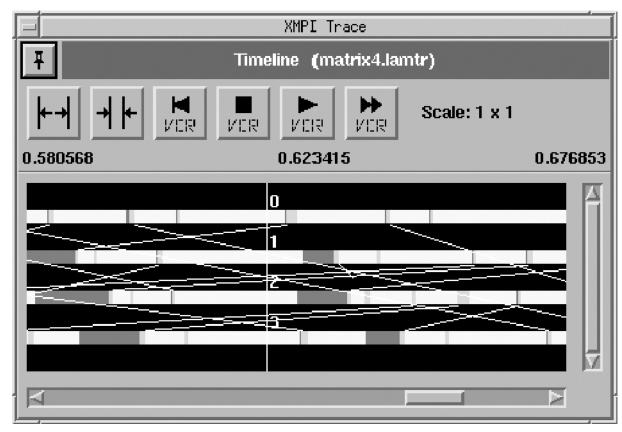


Figura 2. XMPI

OpenMosix utiliza un archivo de configuración que se encuentra generalmente en /etc (ver documentación por antiguas versiones de este archivo) y se llama openmosix.map, y debe estar en cada nodo. Su formato es muy simple y cada línea tiene tres campos: Nodo_ID IP-Address(o hostname) Range-size

Un ejemplo sería:

```
1 node1 1
2 node2 1
3 node3 1
4 192.168.1.1 1
5 192.168.1.2 1
```

También se puede utilizar un rango donde el ID y la IP se incrementan respectivamente. Hay que procurar tener en cada nodo la misma configuración y la misma versión de OpenMosix. Para ejecutar OpenMosix, se debe hacer en cada nodo:

```
setpe -w -f /etc/openmosix.map
```

También se puede utilizar el *script* OpenMosix (copiándolo de userspace-tools a /etc/init.d) para arrancarlo durante el *boot*.

El sistema de archivos oMFS permite el acceso remoto a todos los archivos en el *cluster*, como si ellos estuvieran localmente montados. Los sistemas de archivos (FS) de sus otros nodos pueden ser montados sobre /mfs y, por lo tanto, los archivos de /home sobre el nodo 3 se verán en cada máquina en /mfs/3/home.

Todos los UIDs (User IDs) y GIDs (Group IDs) del FS sobre cada nodo del *cluster* deberán ser iguales (podría utilizarse OpenLdap para este fin).

Para montar el oMFS, se deberá modificar /etc/fstab con una entrada como: mfs_mnt /mfs mfs dfsa = 1 0 0 y para habilitarlo o para inhibirlo: mfs_mnt /mfs mfs dfsa = 0 0 0.

Después, el FS de cada nodo se verá en mfs/[openMosixNode ID]/. Una vez instalado, se podría ejecutar varias veces un *script* muy simple como, por ejemplo (ver *Howto* de OpenMosix):

```
awk 'BEGIN {for(i = 0;i<10000;i++)for(j = 0;j<10000;j++);}'
```

Y, a continuación, observar el comportamiento con mosmom o con openmosixview (recomendado). OpenMosix posee un *daemon* (omdiscd), que permite configurar automáticamente el *cluster* eliminando la necesidad de editar y configurar /etc/openmosix.map. Este *daemon* utiliza multicast para indicarles a los otros nodos que él también es un nodo OpenMosix, por lo cual, una vez arrancado el omdiscd, este *daemon* se unirá al *cluster* de forma automática. Para ello, es necesario tener el *default routing* (GW) de la red bien configurado. Una vez ejecutado (omdiscd), se generarán una serie de mensajes que indicarán el estado del *cluster* y la configuración. Con el comando showmap se podrá ver la nueva configuración generada por omdiscd. OpenMosix provee un conjunto de herramientas para que el administrador pueda configurar y sintoniza el *cluster* OpenMosix. Estas tareas se pueden realizar con herramientas en el espacio de usuario (*migrate*, *mon*, *mosctl*, *mosrun*) o a través de la interfaz /proc/hpc. Es importante tener en cuenta que hasta OpenMosix versión 2.4.16 la interfaz se llamaba /proc/mosix y desde la versión 2.4.17 se llama /proc/hpc.

A continuación, se presenta un resumen de las herramientas de configuración que se ejecutan en espacio de usuario; para las /proc/hpc consultar las referencias:

- migrate [PID] [OpenMosix ID]: envía una petición de migración a un proceso.

- mon: es un monitor con interfaz de texto que muestra información sobre el *cluster* a través de un diagrama de barras.

- mosctl: es la herramienta de configuración de OpenMosix. A través de las opciones (stay, lstay, block, quiet, mfs, expel, bring, get- tune, getyard, getdecay) se le puede indicar a los procesos si pueden migrar o no, la utilización MFS, obtener información sobre la carga, equilibrio de la misma, etc.

- mosrun [h | OpenMosix ID | list of OpenMosix IDs] command [arguments]: ejecuta un comando sobre un nodo determinado.

3. *Metacomputers, grid computing*

Los requisitos de cómputo necesarios para ciertas aplicaciones son tan grandes que requieren miles de horas para poder ejecutarse en entornos de *clusters*. Tales aplicaciones han promovido la generación de ordenadores virtuales en red, *metacomputers* o *grid computers*. Esta tecnología ha permitido conectar entornos de ejecución, redes de alta velocidad, bases de datos, instrumentos, etc., distribuidos geográficamente. Esto permite obtener una potencia de procesamiento que no sería económicamente posible de otra manera y con excelentes resultados. Ejemplos de su aplicación son experimentos como el I-WAY networking (el cual conecta superordenadores de 17 sitios diferentes) en América del Norte, o DataGrid, CrossGrid en Europa o IrisGrid en España. Estos *metacomputers o grid computers* tienen mucho en común con los sistemas paralelos y distribuidos (SPD), pero también difieren en aspectos importantes. Si bien están conectados por redes, éstas pueden ser de diferentes características, no se puede asegurar el servicio y están localizadas en dominios diferentes. El modelo de programación y las interfaces deben ser radicalmente diferentes (con respecto a la de los sistemas distribuidos) y adecuadas para el cómputo de altas prestaciones. Al igual que en SPD, las aplicaciones de *metacomputing* requieren una planificación de las comunicaciones para lograr las prestaciones deseadas; pero dada su naturaleza dinámica, son necesarias nuevas herramientas y técnicas. Es decir, mientras que el *metacomputing* puede formarse con la base de los SPD, para éstos es necesario la creación de nuevas herramientas, mecanismos y técnicas. [Fos]

3.1. Diferentes arquitecturas de cómputo

Si se tiene en cuenta sólo el aspecto de potencia de cálculo, podemos ver que existen diversas soluciones en función del tamaño y las características del problema. En primer lugar, se podría pensar en un superordenador (servidor), pero presentan problemas como falta de escalabilidad, equipos y mantenimiento costoso, cómputo de picos (pasan mucho tiempo desaprovechados) y problemas de fiabilidad. La alternativa económica es un conjunto de ordenadores interconectados por una red de altas prestaciones (Fast Ethernet -LAN- o Myrinet -SAN) lo cual formaría un *cluster* de estaciones dedicado a computación paralela/distribuida (SPD) con un rendimiento muy alto (relación coste/rendimiento 3 a 15 veces). Pero estos sistemas presentan inconvenientes tales como coste elevado de las comunicaciones, mantenimiento, modelo de programación, etc. Sin embargo, es una solución excelente para aplicaciones de grano medio o HTC, *high time computing* ('computación de alta productividad'). Otro concepto interesante es el de *intranet computing,* que significa la utilización de los equipos de una red local (por ejemplo, una red de clase C)

para ejecutar trabajos secuenciales o paralelos con la ayuda de una herramienta de administración y carga; es decir, es el siguiente paso a un *cluster* y permite la explotación de potencia computacional distribuida en una gran red local con las consiguientes ventajas, al aumentar el aprovechamiento de los recursos (ciclos de CPU a bajo coste), mejora de la escalabilidad y administración no demasiado compleja. Para este tipo de soluciones existe software tal como Sun Grid Engine de Sun Microsystems [Sun], Condor de la Universidad de Wisconsin (ambos gratuitos) [Uni] o LSF de Platform Computing (comercial) [Pla].

La opción del *intranet computing* presenta algunos inconvenientes tales como la imposibilidad de gestionar recursos fuera del dominio de administración. Algunas de las herramientas mencionadas (Condor, LSF o SGE) permiten la colaboración entre diferentes subnodos del sistema, pero todos ellos deben tener la misma estructura administrativa, las mismas políticas de seguridad y la misma filosofía en la gestión de recursos. Si bien presenta un paso adelante en la obtención de cómputo a bajo precio, sólo gestionan la CPU y no los datos compartidos entre los subnodos. Además, los protocolos e interfaces son propietarios y no están basados en ningún estándar abierto, no se pueden amortizar los recursos cuando están desaprovechados ni se puede compartir recurso con otras organizaciones. [Beo, Ext, Die]

El crecimiento de los ordenadores entre 1986 y 2000 se ha multiplicado por 500 y las redes por 340.000, pero las predicciones auguran que entre los años 2001 y 2010 los computadores se multiplicarán sólo por 60, y en cambio las redes, por 4.000. Esto indica la pauta de la siguiente arquitectura para HPC: cómputo distribuido en Internet *o grid computing* (GC) *o metacomputing*.

Por lo tanto, *grid computing* es una nueva tecnología emergente, cuyo objetivo es compartir recursos en Internet de manera uniforme, transparente, segura, eficiente y fiable. Esta tecnología es complementaria a las anteriores, ya que permite interconectar recursos en diferentes dominios de administración respetando sus políticas internas de seguridad y su software de gestión de recursos en la intranet. Según unos de sus precursores, Ian Foster, en su artículo "What is the Grid? A Three Point Checklist" (2002), un *grid* es un sistema que:

1) coordina recursos que no están sujetos a un control centralizado,

2) utiliza protocolos e interfaces estándares, abiertas y de propósitos generales, y

3) genera calidades de servicio no triviales.

Entre los beneficios que presenta esta nueva tecnología, se pueden destacar el alquiler de recursos, la amortización de recursos propios, gran potencia sin necesidad de invertir en recursos e instalaciones, colaboración/compartición entre instituciones y organizaciones virtuales, etc.

La figura siguiente da una visión de todos estos conceptos. [Llo]

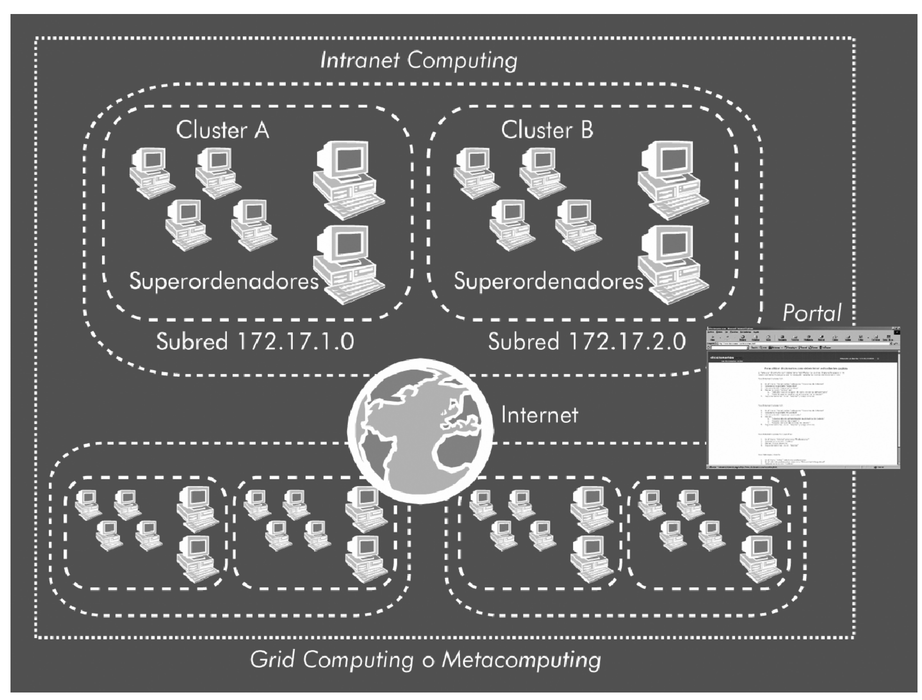


Figura 3

3.2. Globus

El proyecto Globus [Gloa, Glob] es uno de los más representativos en este sentido, ya que es el precursor en el desarrollo de un *toolkit* para el *metacomputing o grid computing* y que proporciona avances considerables en el área de la comunicación, información, localización y planificación de recursos, autentificación y acceso a los datos. Es decir, Globus permite compartir recursos localizados en diferentes dominios de administración, con diferentes políticas de seguridad y gestión de recursos y está formado por un paquete software (*middleware*), que incluye un conjunto de bibliotecas, servicios y API.

La herramienta *globus* (*Globus toolkit*) está formada por un conjunto de módulos con interfaces bien definidas para interactuar con otros módulos y/o servicios. La funcionalidad de estos módulos es la siguiente:

- Localización y asignación de recursos: permite comunicar a las aplicaciones cuáles son los requisitos y ubicar los recursos que los satisfagan, ya que una aplicación no puede saber dónde se encuentran los recursos sobre los cuales se ejecutará.

- Comunicaciones: provee de los mecanismos básicos de comunicación, que representan un aspecto importante del sistema, ya que deben permitir diversos métodos para que se pueda utilizar eficientemente por las aplicaciones. Entre ellos se incluyen paso de mensajes (*message passing*), llamadas a procedimientos remotos (RPC), memoria compartida distribuida, flujo de datos (*stream-based*) y *multicast*.

- Servicio de información (*unified resource information service*): provee un mecanismo uniforme para obtener información en tiempo real sobre el estado y la estructura del metasistema donde se están ejecutando las aplicaciones.

- Interfaz de autentificación: son los mecanismos básicos de autentificación para validar la identidad de los usuarios y los recursos. El módulo genera la capa superior que luego utilizarán los servicios locales para acceder a los datos y los recursos del sistema.

- Creación y ejecución de procesos: utilizado para iniciar la ejecución de las tareas que han sido asignadas a los recursos pasándoles los parámetros de ejecución y controlando éstas hasta su finalización.

- Acceso a datos: es el responsable de proveer un acceso de alta velocidad a los datos almacenados en archivos. Para DB, utiliza tecnología de acceso distribuido o a través de CORBA y es capaz de obtener prestaciones óptimas cuando accede a sistemas de archivos paralelos o dispositivos de entrada/salida por red, tales como los HPSS (*high performance storage system*).

La estructura interna de Globus se puede observar en la siguiente figura (http://www.globus.org/toolkit/about.html).

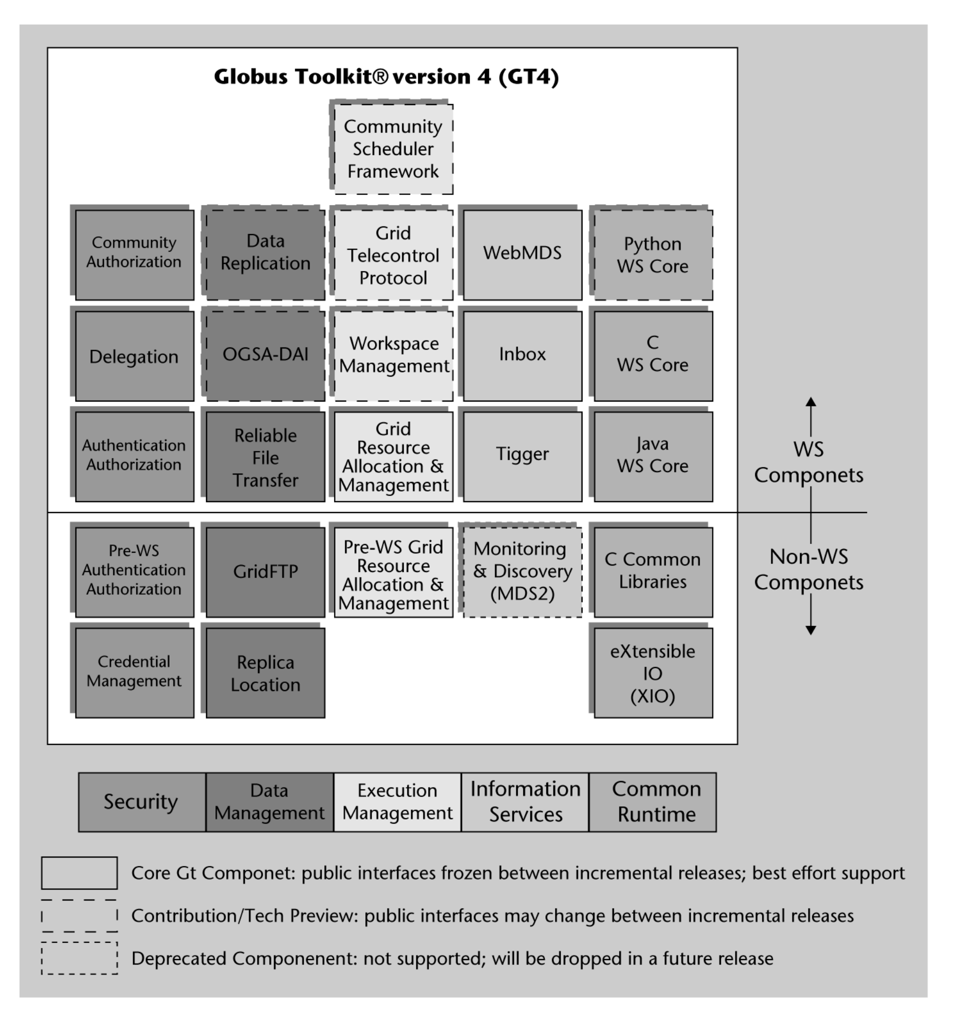


Figura 4

3.3. Software, instalación y administración de Globus

El sitio web de 'The Globus Alliance' es http://www.globus.org [Gloa]. Aquí se pueden encontrar tanto el código fuente, como toda la documentación necesaria para transformar nuestra intranet como parte de un *grid*. Formar parte de un *grid* significa ponerse de acuerdo y adoptar las políticas de todas las instituciones y empresas que forman parte de él. En España existen diferentes iniciativas basadas en Globus. Una de ellas es *IrisGrid* [Llo], a la cual es posible unirse para obtener las ventajas de esta tecnología. Para mayor información, consultar: http://www.rediris.es/irisgrid/.

El primer paso para tener Globus operativo es obtener el software (en este momento es Globus Toolkit 4), llamado GT4. Este software implementa los servicios con una combinación entre C y Java (los componentes C sólo se pueden ejecutar en plataformas UNIX GNU/ Linux generalmente) y es por ello por lo que el software se divide por los servicios que ofrece. En función del sistema que se quiera instalar se deberán obtener unos paquetes u otros.

Una guía de instalación rápida con los pre-requisitos, instalación y generación de certificados se puede obtener desde http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html. En resumen se siguen los siguientes pasos:

1) Pre-requisitos: verificar software y versiones (zlib, j2se, deshabilitar gcj, apache, C/C++, tar, make, sed, perl, sudo, postgres, iodbc)
2) Crear usuario, bajar y compilar GT4
3) Inicializar la seguridad para el sistema (certificados)
4) Inicializar GridFTP
5) Inicializar los Webservices Container
6) Configurar RFT (Reliable File Transfer)
7) Inicializar el WS GRAM (job management)
8) Inicializar la segunda máquina
9) Inicializar los servicios de Index Service hierarchy
10) Inicializar el cluster
11) Establecer Cross-CA Trust

Como se puede observar, no es una tarea fácil instalar y poner a funcionar un GT4, lo cual se justifica para incorporar un cluster a un *grid* o si se quieren hacer unas pruebas (se recomienda una dosis extra grande de ilusión y paciencia) para ver la potencialidad del GT4. Para una información detallada de cómo proceder con la instalación, consultar:

http://www.globus.org/toolkit/docs/4.0/admin/docbook/

Actividades

1) Instalar PVM sobre un nodo y ejecutar el programa master.c y cliente.c dados como ejemplos y observar su comportamiento a través de xpmv.

2) Instalar y configurar Mpich sobre un nodo; compilar y ejecutar el programa cpi.c.

3) Instalar y configurar LAM-MPI sobre un nodo; compilar y ejecutar el programa cpi.c. y observar su comportamiento a través de xmpi.

Otras fuentes de referencia e información

[Debc, Ibi, Mou01]

Lam-mpi: http://www.lam-mpi.org/

System-config-cluster (FC): http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.html

OpenMosix: http://openmosix.sourceforge.net/

HowTo Openmosix: http://howto.x-tend.be/openMosix-HOWTO/

Globus4: http://www.globus.org/toolkit/docs/4.0/

GT4 Quick Guide: http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html

Bibliografía

[Aiv02] **Tigran Aivazian** (2002). "Linux Kernel 2.4 Internals". *The Linux Documentation Project* (guías).

[Ano99] **Anónimo.** *Maximum Linux Security: A Hacker's Guide to Protecting*

[Apa] Apache2 + SSL.
<http://www.debian-administration.org/articles/349>

[Apab] Apache2 + WebDav
<http://www.debian-administration.org/articles/285>

[Apac] Apache2 + Subversion
<http://www.debian-administration.org>

[Ar01] **Jonathan Corbet; Alessandro Rubini**. *Linux Device Drivers 2nd Editon*. O'Reilly, 2001.

[Arc] **Roberto Arcomano**. "Kernel Analysis-HOWTO". *The Linux Documentation Project*.

[Aus] **CERT Australia.** "Australian CERT".
<http://www.auscert.org.au/>

[Bac86] **Maurice J. Bach** (1986). *The Design of the UNIX Operating System*. Prentice Hall.

[Bai03] **Edward C. Bailey** (2003). *RedHat Maximum RPM*.
<http://www.redhat.com/docs/books/max-rpm/index.html>

[Ban] **Tobby Banerjee.** "Linux Installation Strategies HOWTO". *The Linux Documentation Project*.

[Bar] **Barrapunto**. *barrapunto site*.
<http://barrapunto.com>

[Bas] **Mike G.** "BASH Programming - Introduction HOWTO". *The Linux Documentation Project.*

[Beo] **Beowulf.org**. *Beowulf Web Site*.
<http://www.beowulf.org>

[Bor] **Matthew Borowski** (2000). "FTP". *The Linux Documentation Project*.

[Bro] **Scott Bronson** (2001). " VPN PPP-SSH". *The Linux Documentation Project*.

[Bul] **Bulma**. "Bulma Linux User Group".
<http://bulmalug.net>

[Bur02] **Hal Burgiss** (2002). "Security QuickStart HOWTO for Linux". *The Linux Documentation Project*.

[Cac] Monitorización con Cacti.
<http://cacti.net/>

[Cdg] **Cedega**. (Entorno para portabilidad juegos GNU/Linux)
<http://www.transgaming.com/>

[Ced] **Cederqvist**. "Version Management with CVS".
<http://www.cvshome.org>

[Cen] The Comunity ENTerprise Operatyng System
<http://www.centos.org>

[CERa] **CERT**. "CERT site".
<http://www.cert.org>

[CERb] **CERT** (2003). "CERT vulnerabilidades".
<http://www.cert.org/nav/index red.html>

[Cerc] **Cervisia.** "Interfaz Cervisia para CVS".
<http://cervisia.sourceforge.net>

[Cis00] **Cisco** (2000). “TCP/IP White Paper”.
<http://www.cisco.com>

[Com01] **Douglas Comer** (2001). *TCP/IP Principios básicos, protocolos y arquitectura*. Prentice Hall.

[Coo] **Mendel Cooper** (2006). “Advanced bashScripting Guide”. *The Linux Documentation Project* (guías).

[CVS] **CVShome.org**. “CVS Home”.
<http://www.cvshome.org>

[CVSI] Interfaces gráficas para CVS<http://www.twobarleycorns.net/tkcvs.html>

[DBo] **Marco Cesati; Daniel Bovet** (2006). *Understanding the Linux Kernel* (3.ª ed.). O'Reilly.

[Deb] **Debian**. “Sitio Seguridad de Debian”.
<http://www.debian.org/security/>

[Deb04] **Debian** (2004). “APT-HOWTO”.
<http://www.debian.org/doc/manuals/apt-howto/index.en.html>

[Deba] **Debian**. “Software Libre vs Software Abierto”.
<http://www.debian.org/intro/free.es.html>

[Debb] **Comunidad Debian**. “Distribución Debian”.
<http://www.debian.org>

[Dieb] **Hank Dietz** (2004). “Linux Parallel Processing”. *The Linux Documentation Project*.

[Dis] **Distrowatch.** “Distribuciones Linux disponibles”.
<http://www.distrowatch.com>

[Dgn] The Dot Gnu Project.
<http://www.gnu.org/software/dotgnu/>

[DNS] Inicialización de un Servidor DNS.
<http://tldp.org/HOWTO/DNS-HOWTO-7.html>

[Dra] **Joshua Drake** (1999). “Linux Networking”. *The Linux Documentation Project*.

[DSL] **Digital Line Subscriber** (2002). *The Linux Documentation Project*.

[Buy] **Kris Buytaert y otros** (2002). “The OpenMosix”. *The Linux Documentation Project*.

[Ext] **ExtremeLinux.org**. “Extreme Linux Web Site”.
<http://www.extremelinux.org>

[Exim] **Exim**. Servidor de correo (MTA).
<http://www.exim.org/docs.html>

[FBI] FBI. “Brigada del FBI para cibercrimen”.
<http://www.emergency.com/fbi-nccs.htm>

[Fed] The Fedora Project.
<http://fedoraproject.org>

[Fen02] **Kevin Fenzi.** “Linux security HOWTO”. *The Linux Documentation Project*.

[Fos] **Ian Foster; Carl Kesselmany** (2003). “Globus: A Metacomputing Infrastructure Toolkit”.
<http://www.globus.org>

[Fre] **Freshmeat**. “Freshmeat site”.
<http://freshmeat.org>

[Fri02] **Aleen Frisch** (2002). *Essential System Administration*. O'Reilly.

[Fry] Monitorización con Frysk.
<http://sources.redhat.com/frysk/>

[FSF] **FSF.** “Free Software Foundation y Proyecto GNU”.
<http://www.gnu.org>

[Gar98] **Bdale Garbee** (1998). *TCP/IP Tutorial*. N3EUA Inc.

[Gloa] **Globus. GT4.** “Admin Guide Installation” y “Admin Guide Configuration”.
<http://www.globus.org>

[Glob] **Globus**. “User’s Guide Core Framework Globus Toolkit ”,
<http://www.globus.org>

[Gt] **Dirk Allaert Grant Taylor**. “The Linux Printing HOWTO”. *The Linux Documentation Project.*

[GT4] Quick Guide.
<http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html>

[Gnu] **Gnupg.org**. *GnuPG Web Site.*
<http://www.gnupg.org/>

[Gon] **Guido Gonzato**. “From DOS/Windows to Linux HOWTO”. *The Linux Documentation Project.*

[Gor] **Paul Gortmaker** (2003). “The Linux BootPrompt HOWTO”. *The Linux Documentation Project.*

[Gre] **Mark Grennan**. “Firewall and Proxy Server HOWTO”. *The Linux Documentation Project.*

[Hat01] **Brian Hatch** (2001). *Hacking Linux Exposed*. McGraw-Hill.

[Hat03] **Red Hat** (2003). “Firewalls” en Red Hat 9 manual.
<http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide
/ch-fw.html#S1-FIREWALL-IPT>

[Hatb] **Red Hat** (2003). “Red Hat 9 Security Guide”.
<http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide/>

[Hatc] **Red Hat** (2003). “Sitio Seguridad de Red Hat”.
<http://www.redhat.com/security/>

[Hatd] **Red Hat** (2003). *Utilización firmas GPG en Red Hat*.
<http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/ch-gnupg.html>

[Hen03] **Bryan Henderson**. “Linux Loadable Kernel Module HOWTO”. *The Linux Documentation Project.*

[Him01] **Pekka Himanen** (2001). *La ética del hacker y el espíritu de la era de la información*. Destino.

[Hin00] **Martin Hinner**. “Filesystems HOWTO”. *The Linux Documentation Project.*

[His] **HispaLinux**. “Comunidad Hispana de Linux”.
<http://www.hispalinux.es>

[IET] **IETF**. “Repositorio de Request For Comment desarrollados por Internet Engineering Task Force (IETF) en el Network Information Center (NIC)”.
<http://www.cis.ohio-state.edu/rfc/>

[Ian] **Iana**. “Lista de puertos TCP/IP”.
<http://www.iana.org/assignments/port-numbers>

[IP] Encaminamientos con la herramienta ip.
ftp://ftp.inr.ac.ru/ip_routing/>

[ipw] Firmaware para tarjetas wireless IPW2200.
<http://ipw2200.sourceforge.net/firmware.php>

[Ibi] **Ibiblio.org** (2003). “Linux Documentation Center”.
<http://www.ibiblio.org/pub/Linux/docs/HOWTO/>

[Incb] **Incidents.org**. “vulnerabilidades Incidents”.
<http://isc.incidents.org>

[Ins] **Insecure.org** (1998). “Vulnerabilidades y exploits”.
<http://www.insecure.org/sploits.html>

[Insa] **Insecure.org**. “Insecure.org site”.
<http://www.insecure.org>

[Insb] **Insecure.org** (2003). “Nmap”.
<http://www.insecure.org/nmap/index.html>

[Log] LogCheck.
<http://logcheck.org/>

[LWP] LWP: Apache+MySQL+:PHP.
<http://www.lawebdelprogramador.com/temas/tema_stablephpapachemysql.php>

[Joh98] **Michael K. Johnson** (1998). “Linux Information Sheet”. *The Linux Documentation Project*.

[Jou] **Linux Journal.** *Linux Journal [Revista Linux].*
<http://www.linuxjournal.com>

[Kan] **Ivan Kanis.** “Multiboot with GRUB Mini-HOWTO”. *The Linux Documentation Project.*

[Kat] **Jonathan Katz**. “Linux + Windows HOWTO”. *The Linux Documentation Project*.

[KD00] **Olaf Kirch; Terry Dawson**. *Linux Network Administrator's Guide*. O'Reilly Associates.
Y como *e-book* (free) en Free Software Foundation, Inc., 2000.
<http://www.tldp.org/guides.html>

[Ker02] **Kernelhacking.org** (2002). “Kernel Hacking Doc Project”.
<http://www.kernelhacking.org>

[Kera] **Kernelnewbies.org**. “Kernel Newbies”.
<http://www.kernelnewbies.org>

[Kerb] **Kernel.org**. “Linux Kernel Archives”.
<http://www.kernel.org>

[Kie] **Robert Kiesling** (1997). “The RCS (Revision Control System)”. *The Linux Documentation Project*.

[Knp] Distribución Knoppix.
<http://knoppix.org>

[Koe] Kristian Koehntopp. “Linux Partition HOWTO”. *The Linux Documentation Project.*

[Kuk] **Thorsten Kukuk** (2003). “The Linux NIS(YP)/NYS/NIS+”. *The Linux Documentation Project*.

[Lam] **LamMPI.org**. “LAM (Local Area Multicomputer)”.
<http://www.lam-mpi.org>

[Law07] **David Lawyer** (2007). “Linux Módem”. *The Linux Documentation Project*.

[Lev02] **Bozidar Levi** (2002). *UNIX administration*. CRC Press.

[Lev] **Eric Levenez**. “UNIX History”.
<http://www.levenez.com/unix>

[Lin03b] *FHS Standard,* 2003.
<http://www.pathname.com/fhs>

[Linc] *Linux Standards Base project*.
<http://www.linux-foundation.org/en/LSB>

[Line] **Linuxsecurity.com**. *Linux Security Reference Card*.
<http://www.linuxsecurity.com/docs/QuickRefCard.pdf>

[lkm] **lkml.** *Linux Kernel Mailing List*.
<http://www.tux.org/lkml

[Llo] **Ignacio Martín Llorente**. *Estado de la Tecnología Grid y la Iniciativa IrisGrid*.
<http://www.rediris.es/irisgrid>

[Lan] **Nicolai Langfeldt; Jamie Norrish** (2001). "DNS". *The Linux Documentation Project*.

[Log] **Logcheck**. "Logckeck Web Site".
<http://logcheck.org/>

[LPD] **LPD**. *The Linux Documentation Project*.
<http://www.tldp.org>

[Mag] **Linux Magazine**. *Linux Magazine*.
<http://www.linux-mag.com/>

[Maj96] **Amir Majidimehr** (1996). *Optimizing UNIX for Performance*. Prentice Hall.

[Mal96] **Fred Mallett** (1996). *TCP/IP Tutorial*. FAME Computer Education.

[Mal07] **Luiz Ernesto Pinheiro Malère** (2007). "Ldap". *The Linux Documentation Project*.

[Miq] **Miquel, S.** "NIS Debian". *Sobre Debian Woody*, /usr/doc/nis/ nis.debian.howto.

[Moin] Moin Moin
<http://moinmoin.wikiwikiweb.de/>

[Moi] Moin Moin + Debian.
<http://moinmoin.wikiwikiweb.de/MoinMoinPackages/DebianLinux>

[Mon] Monit.
<http://www.tildeslash.com/monit/>

[Monb] Monitorización con Munin y monit.
<http://www.howtoforge.com/server_monitoring_monit_munin>

[Monc] Monitorización con SNMP y MRTG.
<http://www.linuxhomenetworking.com/wiki/index.php/>
Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance

[Mono] Mono project.
<http://www.mono-project.com/Main_Page>

[Mor03] **Daniel Morill** (2003). *Configuración de sistemas Linux*. Anaya Multimedia.

[Mou01] **Gerhard Mourani** (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.

[Mun] Munin.
<http://munin.projects.linpro.no/>

[MRTG] MRTG.
<http://oss.oetiker.ch/mrtg/>

[Mur] **Gary Lawrence Murphy**. *Kernel Book Project*.
<http://kernelbook.sourceforge.net>

[Mutt] Cliente de correo Mutt.
<http://www.mutt.org>

[Mys] **Mysql**. "Reference Manual".
<http://www.mysql.com/>

[MysqlA] **Mysql Administrator.**
<http://www.mysql.com/products/tools/administrator/>

[Nes] **Nessus.org**. "Nessus".
<http://www.nessus.org>

[Net] **Netfilter.org**. *Proyecto Netfilter/IPtables*.
<www.netfilter.org>

[Neu] **Christopher Neufeld**. "Setting Up Your New Domain Mini-HOWTO". *The Linux Documentation Project.*

[New] **Newsforge**. "Newsforge site".
<http://newsforge.org>

[NIS] Inicialización de un Servidor NIS.
<http://tldp.org/HOWTO/NIS-HOWTO/verification.html>

[NSAa] **NSA**. "NIST site".
<http://csrc.nist.gov/>

[NSAb] **NSA** (2003). "Security Enhanced Linux".
<http://www.nsa.gov/selinux>

[Nt3] NTFS-3g Project: NTFS-3G Read/Write Driver.
<http://www.ntfs-3g.org/>

[Oke] **Greg O'Keefe**. "From Power Up To bash Prompt HOWTO". *The Linux Documentation Project*.

[Open] **OpenVPN.** Red privada virtual.
<http://openvpn.net/howto.html>

[OpenM] OpenMosix.
<http://openmosix.sourceforge.net/>

[OpenMb] HowTo Openmosix.
<http://howto.x-tend.be/openMosix-HOWTO/>

[OSDa] **OSDL**. "Open Source Developement Laboratories".
<http://www.osdl.org>

[OSDb]OSDN. "Open Source Development Network".
<http://osdn.com>

[OSIa] **OSI**. "Listado de licencias Open Source".
<http://www.opensource.org/licenses/index.html>

[OSIb] **OSI** (2003). "Open Source Definition".
<http://www.opensource.org/docs/definition.php>

[OSIc] **OSI** (2003). "Open Source Iniciative".
<http://www.opensource.org>

[Peñ] **Javier Fernández-Sanguino Peña** (2007). "Securing Debian Manual".
<http://www.debian.org/doc/manuals/securing-debian-howto/>

[Pga] **PgAccess**. Cliente para PostgreSQL.
<http://www.pgaccess.org/>

[Pla] **Plataform**. "LSF".
<http://www.platform.com>

[Posa] **PostgreSQL.org**. "PostgreSQL Administrator's Guide".
<http://www.postgresql.org/docs/>

[Per] Performance Monitoring Tools for Linux.
<http://www.linuxjournal.com/article.php?sid=2396>

[Pose] **PostgreSQL.org**. "PostgreSQL Web Site".
<http://www.postgresql.org>

[PPP] **Linux PPP** (2000). "Corwin Williams, Joshua Drake and Robert Hart". *The Linux Documentation Project*.

[Pra03] **Joseh Pranevich** (2003). "The Wonderful World of Linux 2.6".
<http://www.kniggit.net/wwol26.html>

[Pri] **Steven Pritchard**. "Linux Hardware HOWTO". *The Linux Documentation Project.*

[Pro] **GNU Project**. "Grub Manual".
<http://www.gnu.org/software/grub/manual/>

[Proa] **Bastille Project**. "Bastille".
<http://bastille-linux.sourceforge.net/>

[Prob] **Mpich Project**. "MPI".
<http://www.mcs.anl.gov:80/mpi/>

[Proc] **Mpich Project**. "Mpich MPI Freeware".
<http://www-unix.mcs.anl.gov/mpi/>

[Prod] **OpenMosix Project**. "OpenMosix".
<http://openMosix.sourceforge.net>

[Proe] **PVM Project**. "PVM Web Site".
<http://www.csm.ornl.gov/pvm/>

[Proc] ProcMail.
<http://www.debian-administration.org/articles/242>

[ProX]Proxy Cache.
<http://www.squid-cache.org/>

[ProT] Proxy Transparente.
<http://tldp.org/HOWTO/TransparentProxy-1.html>

[Prof] ProFTP: Servidor de archivos por FTP.
<http://www.debian-administration.org/articles/228>

[PS02] **Ricardo Enríquez Pio Sierra** (2002). *Open Source*. Anaya Multimedia.

[PurF] PureFTP: Servidor de archivos por FTP.
<http://www.debian-administration.org/articles/383>

[Qui01] **Ellie Quigley** (2001). *Linux shells by Example*. Prentice Hall.

[Ran] **David Ranch** (2005). "Linux IP Masquerade" y **John Tapsell**. *Masquerading Made Simple*. The Linux Documentation Project.

[Ray98] **Eric Raymond** (1998). "La catedral y el bazar".
<http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html>

[Ray02a] **Eric Raymond** (2002). "UNIX and Internet Fundamentals". *The LinuxDocumentation Project*.

[Rayb] **Eric Steven Raymond**. "The Linux Installation HOWTO". *The Linux Documentation Project*.

[Rad] **Jacek Radajewski; Douglas Eadline** (2002). "Beowulf: Installation and Administration". En: Kurt Swendson. *Beowulf HOWTO (tlpd)*.
<http://www.sci.usq.edu.au/staff/jacek/beowulf>

[Red] Optimización de servidores Linux.
<http://people.redhat.com/alikins/system_tuning.html>

[Redb] System-config-cluster (FC).
<http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/
Cluster_Administration/index.html>

[Redh] Red Hat Inc. "Distribución Red Hat".
<http://www.redhat.com>

[Rid] **Daniel Lopez Ridruejo** (2000). "The Linux Networking Overview". *The Linux Documentation Project*.

[Rus] **Rusty Russell**. "Linux IPCHAINS". *The Linux Documentation Project*.

[SM02] **Michael Schwartz y otros** (2002). *Multitool Linux - Practical Uses for Open Source Software*. Addison Wesley.

[Sal94] **Peter H. Salus** (1994). “25 aniversario de UNIX” (núm.1, noviembre). *Byte España*.

[Sam] Samba Project.
<http://samba.org>

[Sama] Samba HOWTO and Reference Guide (Chapter Domain Control).
<http://samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html>

[Samb] Samba Guide (Chapter Adding Domain member Servers and Clients).
<http://samba.org/samba/docs/man/Samba-Guide/unixclients.html>

[San] **Sans**. “Top20 de vulnerabilidades”.
<http://www.sans.org/top20/>

[Sci] Scientific Linux.
<http://www.scientificlinux.org>

[Sec] **Andrés Seco** (2000). “Diald”. *The Linux Documentation Project*.

[Sei02] **Kurt Seifried** (2002). “Securing Linux, Step by Step”.
<http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html>

[Skoa] **Miroslav Skoric**. “LILO mini-HOWTO”. *The Linux Documentation Project*.

[Skob] **Miroslav Skoric.** “Linux+WindowsNT mini-HOWTO”. *The Linux Documentation Project*.

[Sla] **Slashdot.** “Slashdot site”.
<http://slashdot.org>

[Smb] Entrada “Server Message Block” a la wikipedia.
<http://en.wikipedia.org/wiki/Server_Message_Block>

[Smi02] **Rod Smith** (2002). *Advanced Linux Networking*. Addison Wesley.

[Sno] **Snort.org**. *Snort*.
<http://www.snort.org>

[Sou] **Sourceforge**. “Sourceforge site”.
<http://sourceforge.org>

[Squ] Squid proxy server.
<http://www.squid-cache.org/>

[Sta02] **Richard Stallman** (2002). “Discusión por Richard Stallman sobre la relación de GNU y Linux”.
<http://www.gnu.org/gnu/linux-and-gnu.html>

[Stu] **Michael Stutz**. “The Linux Cookbook: Tips and Techniques for Everyday Use”. *The Linux Documentation Project* (guías).

[Ste07] Steve French, Linux CIFS Client guide.
<http://us1.samba.org/samba/ftp/cifs-cvs/linux-cifs-client-guide.pdf>

[Stei] **Tony Steidler-Dennison** (2005). Your Linux Server and Network. Sams.

[Sub] Subversion.
<http://subversion.tigris.org>

[Subb] Control de versiones con Subversion. Free Book.
<http://svnbook.red-bean.com/index.es.html>

*****[Sun02] **Rahul Sundaram** (2002). “The dosemu HOWTO”. *The Linux Documentation Project*.

[Sun] **Sun**. “Sun Grid Engine”.
<http://www.sun.com/software/gridware/>

[Tan87] **Andrew Tanenbaum** (1987). *Sistemas operativos: Diseño e Implementación*. Prentice Hall.

[Tan06] **Andrew Tanenbaum; Albert S. Woodhull** (2006). *The Minix Book: Operating Systems Design and Implementation* (3.ª ed.). Prentice Hall.

[Tkc] **Tkcvs** (2003). “Interfaz Tkcvs para CVS”.
<http://www.tkcvs.org>
<http://www.twobarleycorns.net/tkcvs.html

[Tri] **Tripwire.com**. *Tripwire Web Site*.
<http://www.tripwire.com/>

[Tum02] **Enkh Tumenbayar** (2002). “Linux SMP HOWTO”. *The Linux Documentation Project*.

[Ubn] Distribución Ubuntu.
<http://www.ubuntu.com>

[Uni] **Wisconsin University** (2003). *Condor Web Site*.
<http://www.cs.wisc.edu/condor>

[USA] **Dep. Justicia USA**. “Division del Departamento de justicia de USA para el cibercrimen”.
<http://www.usdoj.gov/criminal/cybercrime/>

[Vah96] **Uresh Vahalia** (1996). *UNIX Internals: The New Frontiers*. Prentice Hall.

[Vas] **Alavoor Vasudevan** (2000). “Modem-Dialup-NT”. *The Linux Documentation Project*.

[Vasa] **Alavoor Vasudevan** (2003). “CVS-RCS (Source Code Control System)”. *The Linux Documentation Project*.

[Vasb] **Alavoor Vasudevan**. “The Linux Kernel HOWTO”. *The Linux Documentation Project*.

[Wm02] **Matt Welsh y otros** (2002). *Running Linux 4th edition*. O'Reilly.

[War] **Ian Ward**. “Debian and Windows Shared Printing mini-HOWTO”. *The Linux Documentation Project*.

[Web] **Webmin**. *Herramienta para la administración de sistemas Linux*.
<http://www.webmin.com/>

[Wil02] **Matthew D. Wilson** (2002). “VPN”. *The Linux Documentation Project*.

[Win] Wine Project.
<http://www.winehq.com/ [Wir] WireShark.>
<http://www.wireshark.org/download.html>

[Woo] **David Wood**. “SMB HOWTO”. *The Linux Documentation Project*.

[Xin] Xinetd Web Site.
<http://www.xinetd.org/>

[Zan] **Renzo Zanelli**. *Win95 + WinNT + Linux multiboot using LILOmini-HOWTO. The Linux Documentation Project*.

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine- readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard- conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly

have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine- readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material.

If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus

accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page,

then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your op-

tion designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified,

and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number.

Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form.

Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of

the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http:// www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number.

If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or

of any later version that has been published (not as a draft) by the

Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front- Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.